

An Algorithm for Mining High Utility Closed Itemsets and Generators

Jayakrushna Sahoo¹, Ashok Kumar Das², and A. Goswami³

^{1,3}*Department of Mathematics, Indian Institute of Technology, Kharagpur 721 302, India*

²*Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India*

Abstract

Traditional association rule mining based on the support-confidence framework provides the objective measure of the rules that are of interest to users. However, it does not reflect the utility of the rules. To extract non-redundant association rules in support-confidence framework frequent closed itemsets and their generators play an important role. To extract non-redundant association rules among high utility itemsets, high utility closed itemsets (HUCI) and their generators should be extracted in order to apply traditional support-confidence framework. However, no efficient method exists at present for mining HUCIs with their generators. This paper addresses this issue. A post-processing algorithm, called the HUCI-Miner, is proposed to mine HUCIs with their generators. The proposed algorithm is implemented using both synthetic and real datasets.

Keywords: Data mining, High utility itemset mining, Association rule mining, Condensed representations, non-redundant association rule.

1 Introduction

Data mining techniques are useful in order to discover efficiently the hidden interesting and useful information from large databases, where the implication of interesting and useful information depends on the problem formulation and the application domain. An important data mining task that has received considerable research attention in recent years is the discovery of association rules from the transactional databases [1, 4, 10, 16, 21, 30, 32, 34, 37, 48, 52, 56]. The traditional association rules mining (ARM) techniques depend on support confidence framework in which all items are given same importance by considering the presence of an item within a transaction, but not the profit of item in that transaction. The goal of such techniques is to extract all the frequent itemsets, then generate all the valid association rules $A \rightarrow B$ from frequent itemset $A \cup B$ whose confidence has at least the user defined confidence threshold. In other words, given a subset of the items in an itemset, we need to predict the probability of the purchase of the remaining items in a transactional database. Nevertheless, this support-confidence framework does not provide the semantic measure of the rule but only it provides the statistical measure as the relative importance of items is not considered. However, such measure is not an adequate measure to the decision maker as the itemset cannot be measured in terms of stock, cost or profit, called utility.

In order to address the above shortcoming of support confidence framework, several researchers have focused on weighted association rule [7, 20, 33, 36, 40, 44, 47, 55]. In such framework, the weights of items (the importance of items to the user) are considered and it also varies differently in application

domains. However, this framework has two pitfalls. Firstly, these schemes still consider the support of an itemset to measure their importance and secondly, these models do not employ the quantities or prices of items purchased. Several researchers have also proposed itemset share measure, which is the fraction of some numerical value in order to overcome these shortcomings [5, 6, 8, 17, 18, 19, 24, 25]. Carter et al. [8] proposed a share-confidence model to discover association rule among numerical attributes which are associated with items in a transaction. Carter et al.'s share-confidence model deals with the amount-share that is a fraction of total weight but not the utility value, such as the net profit, total cost [15]. As a result, this model does not accomplish to conventional utility mining [2, 3, 11, 22, 23, 26, 27, 28, 29, 39, 45, 46, 53, 54] in which the requirements of decision makers are used to extract the itemsets with high utility, the utility of itemset is no less than the user specified minimum utility threshold, which are composed of weights and purchased quantities. The weight represents the importance of distinct items known as *external utility*, and the purchased quantity in each transaction is known as *internal utility* of the items. The product of external utility with sum total of internal utility of an item is called *utility* of the item. As utility does not satisfy *downward closure property* [1], most of the methods proposed in the literature are applied to find the candidate high utility itemsets first and then to identify high utility itemsets by an additional database scan. Some researchers proposed methods to find high utility itemsets without candidate generations [14, 28].

The traditional support-confidence model is limited by the number of rules generated that are often not interesting to the user especially when the confidence threshold is very low and some interesting rules are missed when the confidence threshold is very high. In the extracted rule set, most of the rules share the same semantic measure or statistical measure with other rules, and they are called the redundant rules. Henceforth, it limits the usefulness of the rule set for the user to validate and take decisions. To discover the non-redundant association rules in support-confidence framework, several approaches have been proposed in the literature [4, 10, 21, 31, 32, 37, 41, 51, 52, 56]. In general, the frequent closed itemsets [31] and frequent generators [56] are used to discover non-redundant association rules. However, these methods are developed for support-confidence framework. Therefore, in this paper we raise an important question as follows. How can we compress the association rules in high utility itemset mining? We aim to answer this question by integrating the concept of frequent closed itemsets and frequent generator to high utility itemset. To represent high utility itemset compactly, Chan et al. [9] introduced the concept of utility frequent closed patterns, where the notion of high utility itemset is different from that of the conventional utility mining. Later, Shie et al. [38] proposed the maximal high utility itemset, and Wu et al. [49, 50] proposed the high utility closed itemset. In these works, there is no algorithm was provided to generate rules. So, if we apply traditional association rule mining, the rules generated from these high utility itemsets would contain redundancy, and also of huge size as they compactly represent high utility itemsets but not the rules.

In order to apply the condensed representation of association rules in support-confidence framework to high utility itemset mining, we first integrate the concept of minimal generators of support-confidence framework to the high utility mining. To extract high utility closed itemsets with their generators simultaneously an algorithm named *HUCI-Miner* (High Utility Closed Itemset-Miner) algorithm has been proposed. From the generated high utility closed itemsets with their generators, various condensed representations of association rules from the literature can be extracted from high utility itemsets. Experiments are carried out to show the compactness of the extracted high utility itemsets.

The rest of the paper is organised as follows. In Section 2, we briefly review the existing works proposed in the literature. In Section 3, we provide some basic preliminaries. Section 4 introduces our proposed HUCI-Miner algorithm. We also discuss the procedure for deriving high utility closed itemsets and generators. In Section 5, we simulate the proposed algorithms method in real and synthetic datasets. Finally, the paper ends with the concluding remarks in Section 6.

2 Related work

In this section, we review the existing methods for high utility itemset mining and generation of non-redundant association rules in support-confidence framework. In the support-confidence framework, the non-redundant association rules are generated from the frequent closed itemsets and their generators. We also review existing works on generation of frequent closed itemsets together with their generators as it is essential for generation of non-redundant association rules.

2.1 High utility itemset mining

The traditional association rule mining methods [1] are based on support-confidence framework, where all items are considered with the same level of importance. The methods proposed in [1, 16, 30, 34] to extract association rule follow this classical statistical measurement producing the same result on a given minimum support and minimum confidence. The weighted association rule mining (WARM) generalizes the traditional framework by giving importance to items, where importance is given as weights. Ramkumar et al. [36] introduced the concept of weighted support of itemsets and weighted association rules on the basis of costs assigned to both items and transactions. Later, considering only the item weights into account, Cai et al. [7] proposed the weighted support of association rules. However, the weighted support of the association rules does not satisfy the downward closure property, which results in the performance degradation. In order to overcome such problem, by considering transaction weight, Tao et al. [44] provided the concept of weighted downward closure property. Considering both support and weight of itemsets, Yun [55] then presented a new strategy, called the weighted interesting pattern mining (WIP). Pears et al. [33] further proposed a WARM method that automates the process of weight assignment to the items by formulating a linear model.

In WARM framework, note that the quantities of items in transactions are not considered. Considering items' quantities in transactions and their individual importance, high utility itemset mining (HUIM) received a considerable research attention [2, 3, 11, 14, 22, 23, 26, 27, 28, 29, 39, 45, 46, 53, 54]. Yao et al. [53, 54] proposed a mathematical model of utility mining by generalizing the share-confidence model [5]. As utility mining does not fulfill the *downward closure property*, Liu et al. [29] proposed the two-phase algorithm that uses the *transaction-weighted downward closure property* to prune the candidate high utility itemsets in the first phase and then all the complete sets of high utility itemsets are obtained in the second phase. To reduce the number of candidate itemsets in the first phase, Li et al. [26] also proposed an isolated items discarding strategy (IIDS) to the level-wise utility mining method.

Ahmed et al. [3] proposed a FP-Growth based algorithm [16] that uses a tree structure, named IHUP-Tree, and efficiently generates the candidate high utility itemsets for incremental and the interactive mining. To further reduce the number of itemsets in the first phase, Tseng et al. [45, 46] proposed the tree-based methods, named the UP-Growth and UP-Growth⁺, which use several strategies to decrease the estimated utility value of an itemset, and as a result, they enhance the performance. To avoid the level wise candidate generation and test strategy, Song et al. [39] proposed a concurrent algorithm, called the CHUI-Mine, for mining HUIs from transaction databases using their proposed data structure CHUI-Tree to maintain the information of HUIs. Their proposed algorithm generates the potential high utility itemsets using two concurrent processes: the first process is used for construction and dynamic pruning the tree, and then placing the conditional trees into a buffer, and the second one for reading the conditional pattern list from the buffer and mining HUIs. To speed up the execution and reduce the memory requirement in the mining process, Lan et al. [22] proposed an efficient utility mining approach that adopts a projection-based indexing mechanism that directly generates the required itemsets from the transactions database. Ahmed et al. [2] proposed a novel tree-based candidate pruning technique, called the High Utility Candidates Prune (HUC-Prune), for avoiding more database scans and the level-wise candidate

generation.

To avoid the computational cost of candidate generation and utility computation, Liu and Qu [28] then proposed a data structure, named the utility-list, to store both the utility information about an itemset and the heuristic information for pruning the search space. Using the constructed utility-lists from a mined database, they developed an efficient algorithm, called the HUI-Miner, which mines high utility itemsets without candidate generation in a depth-first search manner. Their algorithm works in a single phase by directly identifying high utility itemsets in an efficient way and it is also scalable. To reduce the cost of join operation in the calculation of the utility-list of an itemset in HUI-Miner, Fournier-Viger et al. [14] improved the HUI-Miner incorporating with the items co-occurrences strategy (named as FHM) that is about six times faster than the HUI-Miner.

2.2 Closed itemsets with their generators and non-redundant association rule mining

To generate both frequent closed itemsets (FCI) and generators, Pasquier et al. [31] proposed the CLOSE algorithm that is based on level-wise searching approach with the help of Apriori property. Szathmary et al. [41] proposed the ZART algorithm that generates FCIs with their generators in a level-wise manner. They further proposed the Eclat-Z algorithm [42] that mines frequent itemsets in a depth-first way and the FCIs with their generators are identified in level-wise manner. An effective method, named as Touch [43], was developed by combining the FCI method Charm [56] and the frequent generator (FG) mining algorithm, Talky-G [43]. The FCIs are mined using Charm and FGs are mined using Talky-G and, then Touch associates the generators to their closed itemsets using a suitable hash function.

Wu et al. [49] introduced the closer concept to high utility itemsets. They called the extracted itemsets as closed⁺ high utility itemsets. On incorporating closure based on support of itemsets, they proved, first mining the set of high utility itemsets and then applying closed constraint produces the same result while mining all the closed itemsets first and then applying the utility constraint. They proposed an effective method named as CHUD (Closed⁺ High Utility itemset Discovery) for mining closed⁺ high utility itemsets. Further, they proposed a method called the DAHU (Derive All High Utility itemsets), to recover all high utility itemsets from the set of closed⁺ high utility itemsets without further accessing the database. In addition, they proposed AprioriHC and AprioriHC-D algorithms [50] and mentioned that CHUD performs better than AprioriHC and AprioriHC-D. However, no suitable method for high utility closed itemset with their generator is proposed for high utility itemset mining.

To reduce the number of association rules extracted in support confidence-framework, several methods have been developed in the literature [4, 10, 21, 31, 32, 37, 41, 51, 52, 56]. Kryszkiewicz [21] proposed the representative association rules (RR) with the help of a cover operator that represents a set of association rules. Zaki [56] proposed a method to reduce the number of association rules and the extracted rules, called the general rules, which have shortest antecedent and shortest consequent giving an equivalence class of rules of same support and confidence. Pasquier et al. [32] defined the min-max rules having minimal antecedent and maximal consequent. Their proposed method eliminates the non-redundant rules as min-max exact and min-max approximate rules from the frequent closed itemsets and their generators. Furthermore, to reduce more rules, Cheng et al. [10] proposed the concept of δ -tolerance, which is a relaxation on the closure defined on the support of frequent itemset. Yahia et al. [52] proposed an informative basis to reduce the number of association rules, which is further efficiently compressed by Sahoo et al. [37]. Xu et al. [51] filtered the min-max rules by defining redundancy and provided the reliable exact basis and reliable approximate basis of the same inference capacity. Balcázar [4] further obtained a small and crisp set of association rules by the help of confidence boost of a rule, which eliminates the rules with similar confidence.

3 Basic preliminaries and problem statement

In this section, we first discuss some basic preliminaries.

Let $I = \{i_1, i_2, i_3, \dots, i_m\}$ be a finite set of items, where each item i_ℓ , $1 \leq \ell \leq m$, have an external utility p_ℓ , $1 \leq \ell \leq m$ in the utility table. A subset $X \subseteq I$ is called an itemset, if X contains k distinct items $\{i_1, i_2, i_3, \dots, i_k\}$, where $i_\ell \in I$, $1 \leq \ell \leq k$, called a k -itemset. Let \mathcal{D} be the task relevant database composed of utility table and the transaction table $T = \{t_1, t_2, t_3, \dots, t_n\}$, containing a set of n transactions, where each transaction $t_d \subseteq I$, $1 \leq d \leq n$, in the database be associated with a unique identifier, say t_{id} . In every transaction t_d , $1 \leq d \leq n$, each item i_ℓ , $1 \leq \ell \leq m$ has a non-negative quantity $q(i_\ell, t_d)$, which represents the purchased quantity known as internal utility of the item i_ℓ in the transaction t_d .

Each itemset X has a statistical measure called the support of X , which is defined by the ratio of the number of transactions containing X to the total number of transactions $|\mathcal{D}|$, and denoted by $supp(X)$. In other words, $supp(X) = \frac{|\{t \in \mathcal{D}, X \subseteq t\}|}{|\mathcal{D}|}$. Let \mathcal{F} be the set of all the itemsets in \mathcal{D} having positive support and $\mathcal{F} = \{X | X \in 2^I, supp(X) > 0\}$. An association rule is an implication of the form $R : X \rightarrow Y$, where $X, Y \subseteq \mathcal{I}$, $Y \neq \emptyset$, and $X \cap Y = \emptyset$. The itemsets X and Y are called antecedent and consequent of the rule R , respectively. Association rules are associated with two statistical measures, which are support and confidence. The support of the rule R is $supp(X \cup Y)$ and the confidence of the rule R is defined by the ratio of the support of $X \cup Y$ to the support of X , and denoted by $conf(R)$. Hence, it is clear that $conf(R) = \frac{supp(X \cup Y)}{supp(X)}$.

We use x_i to denote the i^{th} item of an itemset X .

Definition 1. The utility of an item i_ℓ in a transaction t_d is denoted by $u(i_\ell, t_d)$ and defined by the product of internal utility $q(i_\ell, t_d)$ and external utility p_ℓ of i_ℓ , that is, $u(i_\ell, t_d) = p_\ell \times q(i_\ell, t_d)$.

Definition 2. The utility of an itemset X contained in a transaction t_d , denoted by $u(X, t_d)$ and defined by the sum of utility of every items of X in t_d . In other words, $u(X, t_d) = \sum_{i_\ell \in X \wedge X \subseteq t_d} u(i_\ell, t_d)$.

Definition 3. The utility of an itemset X in \mathcal{D} is denoted by $u(X)$ and defined by the sum of the utilities of X in all the transactions containing X in \mathcal{D} , that is, $u(X) = \sum_{X \subseteq t_d \wedge t_d \in \mathcal{D}} u(X, t_d) = \sum_{X \subseteq t_d \wedge t_d \in \mathcal{D}} \sum_{i_\ell \in X} u(i_\ell, t_d)$.

Definition 4. An itemset X is called a high utility itemset, if the utility of X has at least the user specified minimum utility threshold, min_util . Otherwise, it is called a low utility itemset. Let \mathcal{H} be the complete set of high utility itemsets. Then, $\mathcal{H} = \{X | X \in \mathcal{F}, u(X) \geq min_util\}$.

Example 1. Consider again the transaction database in Table 1 with the utility table given in Table 2. From Table 2, note that the external utility of item B is 4 and the internal utility of the item B in the transaction t_3 is 4. Thus, the utility of item B in t_3 is $u(B, t_3) = p_B \times q(B, t_3) = 4 \times 4 = 16$. The utility of the itemset BF in the transaction t_6 is $u(BF, t_6) = u(B, t_6) + u(F, t_6) = 4 \times 1 + 1 \times 2 = 6$ and the utility of the itemset BF in \mathcal{D} becomes $u(BF) = u(BF, t_3) + u(BF, t_6) = 23$. Note that, if the minimum utility threshold is 20, the itemset BF is a high utility itemset. Table 3 shows all the high utility itemsets.

We say an association rule in high utility itemset mining is valid, if it satisfies the following two conditions:

- (i) The antecedent and itemset formed by combination of antecedent and consequent are high utility itemset.
- (ii) The confidence is more than or equal to the specified minimum confidence threshold, say min_conf .

Table 1: An example transaction database \mathcal{D}

T_{id}	Transaction
t_1	$A(4), C(1), E(6), F(2)$
t_2	$D(1), E(4), F(5)$
t_3	$B(4), D(1), E(5), F(1)$
t_4	$D(1), E(2), F(6)$
t_5	$A(3), C(1), E(1)$
t_6	$B(1), F(2), H(1)$
t_7	$D(1), E(1), F(4), G(1), H(1)$
t_8	$D(7), E(3)$
t_9	$G(10)$

Table 2: Utility table

Item	Utility
A	3
B	4
C	5
D	2
E	1
F	1
G	2
H	1

Table 3: HUIs with minimum utility 20

Itemset	utility	Itemset	utility
A	21	B	20
D	22	G	22
E	22	F	20
AC	31	AE	28
BE	21	DF	24
BF	23	DE	37
FE	36	ACE	38
AFE	20	DFE	36
BDE	23	BFE	22
$ACFE$	25	$BDFE$	24

Generation of valid utility based association rules from high utility itemsets is relatively straightforward. The rules of the form $R : X \rightarrow Y$ are generated for all high utility itemsets X , and $X \cup Y$, for all $X, Y \neq \phi$, and the rule R provides the confidence of the rule having at least min_conf . Since $X \cup Y$ is a high utility itemset, the generated rule is guaranteed to be high utility. To derive all possible valid rules, we need to examine each high utility itemset and repeat the rule generation process as in Apriori algorithm [1] with utility constraint.

4 The proposed HUCI-Miner algorithm

In this section, we first discuss some useful definitions and theorems before describing our proposed algorithm.

The Apriori property used to prune the candidate itemset search space cannot be applied directly to mine high utility itemset, since the utility constraint is neither monotone nor anti-monotone. To reduce the size of search space and enhancing the performance of mining task, Liu et al. [29] proposed the concept of transaction-weighted utility, which satisfies the downward closure property and is based on the following definitions.

Definition 5. The utility of a transaction t_d is denoted by $tu(t_d)$ and defined by $tu(t_d) = u(t_d, t_d)$.

Definition 6. The transaction-weighted utility (TWU) of an itemset X in a database \mathcal{D} is denoted by $twu(X)$ and defined by the sum of the utilities of all the transactions containing X in \mathcal{D} , where $twu(X) = \sum_{t_d \in \mathcal{D} \wedge X \subseteq t_d} tu(t_d)$.

Property 1. The transaction-weighted utility satisfies the downward closure property. That means for a given itemset X , if $twu(X)$ is less than the specified min_util , all supersets of X are not high utility.

Table 4: TWU of each item of database given in Table 1

Item	A	B	C	D	E	F	G	H
TWU	40	31	40	72	112	87	30	17

Table 4 shows the transaction-weighted utility of each item. For example, the transaction utility of t_1 is $tu(t_1) = u(A, t_d) + u(C, t_d) + u(E, t_d) + u(F, t_d) = 12 + 5 + 6 + 2 = 25$. Again, consider the itemset ACE which is in transaction t_1 and t_5 having the transaction-weighted utility of ACE , $twu(ACE) = tu(t_1) + tu(t_5) = 25 + 15 = 40$. If the min_util is set to 45, all supersets of ACE are not high utility itemset according to Property 2. For a given itemset, if its transaction-weighted utility has at least min_util , we call the itemset as high transaction-weighted utility itemset (HTWUI).

Definition 7. An itemset Y is called the closure of an itemset X , denoted by $\gamma(X)$, if there does not exist other large superset of X than Y , with $\text{supp}(X) = \text{supp}(Y)$. An itemset X is then called the closed itemset, if $X = \gamma(X)$.

Property 2. For a given itemset X , $\text{twu}(X) = \text{twu}(\gamma(X))$. In other words, the transaction-weighted utility of an itemset is same as its closure.

Definition 8. The local utility value of an item x_i in an itemset X , denoted by $\text{luv}(x_i, X)$ and defined by the sum of the utility values of the items x_i in all the transactions containing X , that is, $\text{luv}(x_i, X) = \sum_{X \subseteq t_d \wedge t_d \in \mathcal{D}} u(x_i, t_d)$.

Definition 9. The local utility value of an itemset X in another itemset Y such that $X \subseteq Y$, denoted by $\text{luv}(X, Y)$, is the sum of local utility measure values of each item $x_i \in X$ in itemset Y , which is given by $\text{luv}(X, Y) = \sum_{x_i \in X \subseteq Y} \text{luv}(x_i, Y)$.

To calculate the local utility value of an itemset X in another itemset Y such that $X \subseteq Y$, an utility unit array needs to be attached to each high utility itemset, which is defined as follows.

Definition 10. [49] The utility unit array of an itemset $X = \{i_1, i_2, i_3, \dots, i_k\}$ is denoted by $U(X) = \{u_1, u_2, u_3, \dots, u_k\}$, where each u_ℓ is $\text{luv}(i_\ell, X)$, $1 \leq \ell \leq k$.

Example 2. The utility unit array, $U(X)$ of an itemset X contains the local utility values of the constituent items of X . Consider the itemset ACE which appears in transactions t_1 and t_5 in Table 1. The local utility value of the item A in ACE is $\text{luv}(A, ACE) = u(A, t_1) + u(A, t_5) = 21$. The utility unit array of ACE is $U(ACE) = \{21, 10, 7\}$. Further, the local utility value of itemset AE in ACE is $\text{luv}(AE, ACE) = \text{luv}(A, ACE) + \text{luv}(E, ACE) = 28$.

Property 3. [49] For a given itemset X with its utility unit array $U(X)$, the utility of X is defined as $u(X) = \sum_{x_i \in X} \text{luv}(x_i, X)$.

Property 4. [49] The utility and utility unit array of an itemset X can be calculated from the utility unit array of its closure itemset $\gamma(X)$.

Property 5. If an itemset X is a high utility itemset, $\gamma(X)$ is also a high utility itemset. However, the converse is not always true.

4.1 Integrating the closure property with HUIM

In this subsection, we discuss how to unify the minimal generator concept of the traditional ARM into high utility itemset mining. In general, the itemsets in a transactional database are not completely independent from other itemsets. A group of itemsets are common to the same set of transactions, and hence, they have the same support. Using the closure operator the itemsets can be grouped into equivalent classes. Two itemsets in a class are called equivalent if they belong to the same set of transactions. A maximal element in a class, is called the closed itemset, which is the closure of other itemsets in that class, and the minimal elements (smallest subsets of the maximal element of the class) are called the generators. All other elements in a class can be derived using the closed itemsets, and the generators and all the elements in a class have the same support. The closed itemsets with their generators lead to the fundamental principle behind the effective construction of non-redundant association rules in the support-confidence framework.

A natural question arises that how to incorporate the similar strategy in HUIM, which means the formation of high utility closed itemset together with their generators those are also high utility itemsets. As suggested by Wu et al. [49], the closure based on utility of itemset does not achieve a high reduction

on the number of high utility itemset and they defined the closure on the supports of itemsets. On incorporating closure based on support of itemsets, they showed that the join order of closed constraint and utility constraint is commutative. This means, first mine the set of high utility itemsets and then apply closed constraint which will produce the same result while mining all the closed itemsets first and then applying the utility constraint. Concluding that any equivalence class constructed among high utility itemsets using closure based on support, the maximal element can be found in both ways. However, later we show that, this commutativity between utility constraint and closure based on support does not hold for generators. As closed high utility itemsets and high utility closed itemsets are same, so Wu et al. [49] defined the closed+ high utility itemset (CHUI).

Definition 11. An itemset X is called high utility closed itemset, if $X = \gamma(X)$ and $u(X) \geq \text{min_util}$.

4.1.1 Generators with utility constraints

Following the definition of generators of a closed itemset in traditional ARM, the problem arises how we can incorporate it to high utility itemset mining. This can be done in two ways. First mine all generators based on support constraint and then apply utility constraint. Secondly, first find all high utility itemset and then mine all generator applying support constraint. We will analyze this joining order between utility constraint and support constraint to extract the generators and conclude that the result of both ordering are different. In first case, we miss the minimal high utility itemsets of an equivalence class and in the later case we do not lose the minimal high utility itemsets. As a consequence, we suggest finding the minimal generators among high utility itemsets. Thus, the utility constraint should be considered first and then support constraint.

For first composition method, we mine generators based on support and then prune itemsets which do not satisfy the utility constraints. Based on composition order, we can define *generator with high utility* as follows.

Definition 12 (Generator with high utility). An itemset X is a generator with high utility, if there is no proper subset Z of X such that $\text{supp}(Z) = \text{supp}(X)$. Moreover, X must satisfy the utility constraints.

Definition 13 simply states that in order to mine *generator with high utility*, using support constraint, the itemsets are first verified whether they are generators or not. Then the itemsets are tested for high utility itemset with utility constraints. So, if an itemset is not a generator, the itemset is not a *generator with high utility*, without verifying with utility constraints. For example, the itemset $AC(2, 31)$ (itemset(support, utility)) in the transactional database given in Table 1 with minimum utility constraint 20, is not a *generator with high utility* without testing with utility constraint as it is not a generator because there is a subset $A(2, 21)$ whose support is same as support of $AC(2, 31)$. Again, the itemset $AF(1, 14)$ is a generator but not a *generator with high utility* as it does not satisfy the utility constraint of minimum utility 20.

Since the above composition has no restriction on the utility of the subset of generators, so after applying utility constraint some generators are pruned. As all the itemsets of an equivalence class can be generated from both closed itemsets and their generators, if some generators are pruned, some supersets of that itemset may not be generated and result loss of information about high utility itemsets. For example, assume that the minimum utility constraint is 20 in the transactional database shown in Table 1. Consider the equivalence class of the itemset $ACFE(1, 25)$. The generators of this closed itemset are $AF(1, 14)$ and $CF(1, 7)$. After applying the utility constraint, note that there is no generator of $ACFE(1, 25)$ because both $AF(1, 14)$ and $CF(1, 7)$ are pruned as their utility is less than 20. However, from Table 3 we observe that the itemset $AFE(1, 20)$ is a high utility itemset belonging to the equivalence class of $ACFE(1, 25)$. But in the generation process of generators, the itemset $AFE(1, 20)$ is pruned as it is the superset of $AF(1, 14)$ with same support. In other words, before $AF(1, 14)$ is

pruned by the utility constraint, the itemset $AFE(1, 20)$ is pruned by itemset $AF(1, 14)$ as both have same support and later one is the subset of the former. As a result, the generator itemset of $ACFE(1, 25)$ is empty. So, while applying traditional itemset generation procedure from generators of an equivalence class, the itemset $AFE(1, 20)$ will not generate. This problem arises because of considering the closure property first and the utility constraint later.

The other way to join the utility constraints and the closure property is that we need to first mine all the itemsets with utility constraint and then apply closure property to compute the generators. From this ordering, we can define *high utility generators* as follows.

Definition 13 (High utility generators). *An itemset X is a high utility generator, if it is a high utility itemset and there exists no proper high utility subset Z such that $supp(Z) = supp(X)$.*

In this approach after extracting all high utility itemsets, the closure property is applied to compute the generators. Again, from these high utility closed itemsets and their generators, all other high utility itemsets of that equivalent class can be generated using the traditional methods of itemset generation. From the analysis, the results of the joining order between the two constraints are different.

4.1.2 Winding up the discussions

It is well-defined from the analysis, we loose some minimal high utility itemsets, if we first find generators based on support and then apply utility constraint. Nevertheless, in Definition 13, we generate the actual high utility generators. As a consequence, the second approach generates high utility minimal generators of an equivalence class in high utility itemset mining. Throughout this paper we apply the second approach and after onwards we mean the generator means the *high utility generator*.

Assume there is a pre-determined total ordering Ω among the items I in the database \mathcal{D} . Accordingly, if item i is occurred before item j in the ordering, we denote this by $i \prec j$. For $\forall j \in Y$, if $i \prec j$, we say $i \prec Y$, where Y is an itemset. Similarly, for itemsets X and Y if $x_i \prec Y_i$ in accordance to the order relation Ω , for $1 \leq i \leq m$, $m = \min(|X|, |Y|)$, we say $X \prec Y$. This ordering can be used to enumerate all the itemsets without duplication. Hereinafter, we always consider an itemset as an *ordered set*, in particular, it is a sequence of distinct and increasingly sorted items with respect to the TWU values of items. If the TWU values of two items are equal, they are sorted according to the lexicographic order.

Let f be a function that assign to each itemset $X \in \mathcal{H}$ to the set of all transactions that contain X , that is, $f(X) = \{t_d \in T \mid X \subseteq t_d, X \in \mathcal{H}\}$. Clearly, for $X \subset Y$, $f(Y) \subseteq f(X)$. Two itemsets $X, Y \in \mathcal{H}$ are said to be equivalent, denoted by $X \cong Y$, iff $f(X) = f(Y)$. The set of itemsets that are equivalent to an itemset X is denoted by $[X]$ and given by $[X] = \{Y \in \mathcal{H} \mid X \cong Y\}$.

Theorem 1. *Let $X \in \mathcal{H}$. If $luv(X, Y) = u(X)$, then $Y \in [\gamma(X)]$.*

Proof. Since $X \subset Y$, $luv(X, Y) = u(X)$ and $X \in \mathcal{H}$. We then have $Y \in \mathcal{H}$. From Definitions 3, 5 and 6, both X and Y are contained in same transaction. Thus, $f(X) = f(Y) = f(\gamma(X))$ and $Y \cong \gamma(X)$. Hence, $Y \in [\gamma(X)]$. \square

Corollary 1. *If $supp(X) = supp(Y)$ and $X \subseteq Y$, then $luv(X, Y) = u(X)$.*

Corollary 2. *For a given itemset X , if there does not exist any itemset $Y \supset X$ such that $luv(X, Y) = u(X)$, then X is a closed itemset.*

Definition 14. *An itemset $X \in [X]$ is called a generator, if X has no proper subset in $[X]$. In other words, it has no proper subset with the same support and it is a high utility itemset.*

Theorem 2. *Let X be a high utility itemset and $x \in X$. If $X \setminus x$ is a high utility itemset, $X \in [X \setminus x]$ iff $supp(X) = supp(X \setminus x)$.*

Proof. Let $x \in X$ and $X \setminus x$ be a high utility itemset. Let $X \in [X \setminus x]$. Then $f(X) = f(X \setminus x)$ means that $\text{supp}(X) = \text{supp}(X \setminus x)$. The converse follows trivially. \square

Theorem 3. *Let X be a high utility itemset. Then, X is a generator iff $\text{supp}(X) \neq \min\{\text{supp}(X \setminus x) : x \in X, (X \setminus x) \in \mathcal{H}\}$.*

Proof. Let X be a generator. Let g be a high utility itemset of length $k - 1$ with minimum support and a subset of X . Then, $g \subset X \Rightarrow f(g) \supseteq f(X)$. If $f(g) = f(X)$, $\text{supp}(X) = \text{supp}(g)$ and X is not a generator. Moreover, it is not the element with the smallest support, whose closure is $\gamma(X)$. This concludes that $f(g) \supset f(X)$ and hence, $\text{supp}(X) \neq \min\{\text{supp}(X \setminus x) : x \in X, (X \setminus x) \in \mathcal{H}\}$. On the other hand, if $\text{supp}(X) \neq \min\{\text{supp}(X \setminus x) : x \in X, (X \setminus x) \in \mathcal{H}\}$, X is the smallest element of the closure $\gamma(X)$. Hence, X is a generator. \square

Corollary 3. *Let X be a high utility itemset. If X is not a generator, then $\text{supp}(X) = \min\{\text{supp}(X \setminus x) : x \in X, (X \setminus x) \in \mathcal{H}\}$.*

Property 6. *For a high utility closed itemset X , if g be a generator, then $u(g, X) < u(g', X)$, where $g' \in [X]$ and $g \subset g'$.*

Theorem 4. *Let $X \in \mathcal{H}$. The statement “If X is a generator, then $\forall Y \in \mathcal{H}, Y \subset X, Y$ is a generator” is false in this context.*

Proof. This can be proved by giving a counter example. Consider the itemset AFE with support 1 and utility value 20. If the minimum utility is set to 20, this is a generator in the context. However, the subsets AF and AE are not generators. Note that AF is not a high utility itemset, whereas AE is a high utility itemset. \square

Rationale: Theorem 4 states that the subsets of a high utility generator may or may not be a generator. However, in the support-confidence framework the subsets of a generator are generators, and hence, if an itemset is not a generator, its superset is not also a generator. This property is used to pruning the itemsets space to obtain the generators in support-confidence framework. Since this property does not satisfy in the high utility context, this pruning strategy cannot be employed. AF is pruned as it is not a high utility itemset, and AE is pruned as it is not a generator. Thus, by the traditional procedure of a support-confidence framework, the itemset AFE will not be generated. However, it is a minimal itemset in $[AFE]$.

4.2 Deriving high utility closed itemsets and generators

It is observed from the analysis in Section 4.1 that in order to extract all high utility generators, we need to discover the entire space of high utility itemsets. The state-of-the-art algorithm FHM [14] is used to extract all high utility itemsets. The HUCI-Miner algorithm given in Algorithm 1 identifies the high utility closed itemsets and generators among the high utility itemsets. It enables to the efficient generation of the non-redundant association rules among the high utility itemsets. The algorithm outputs the resultant set, CH , which contains the high utility closed itemsets H_k , where each set H_k , $1 \leq k \leq \text{max}$, has all high utility k -itemsets and max is the size of the longest high utility itemset. This algorithm generates the high utility closed itemsets and generators from the high utility itemsets using Theorem 2, 3 and Corollary 3. Note that no additional database scan is required in order to find out the *utility unit array* of each closed itemset, which is used to calculate the local utility value of any subset. By scarifying a little more memory consumption, this can be calculated from the *utility-list* of the constituent items of an itemset.

ALGORITHM 1: *HUCI-Miner*(\mathcal{D}, \min_util)

Input: $HUI : \{H_1, H_2, \dots, H_{max}\}$, where H_k is the set of hui of length k , max is the size of largest high utility itemset.

Output: High utility closed itemsets with their generators.

```
1 CH  $\leftarrow \emptyset$ , HG  $\leftarrow \emptyset$  // HG: Set of high utility generators
2  $H = FHM(\mathcal{D}, \min\_util)$  //  $H = \{H_1, H_2, \dots, H_{max}\}$ ,  $H_k$  Set of high utility  $k$ -itemset
3 for each itemset  $h \in H_1$  do
4    $h.closed \leftarrow true$ ;  $h.key \leftarrow true$ ;
5 end
6 for ( $k = 2; k \leq max; k++$ ) do
7   if  $H_k \neq \emptyset$  then
8     for each itemset  $h \in H_k$  do
9        $h.key \leftarrow true$ ;  $h.closed \leftarrow true$ ;
10      for all subsets  $h' \in H_{k-1}$  of  $h$  do
11        if ( $supp(h') == supp(h)$ ) then
12           $h.key \leftarrow false$ ;  $h'.closed \leftarrow false$ ;
13        end
14      end
15       $CH_{k-1} \leftarrow \{h \in H_{k-1} | h.closed = true\}$ ;
16       $Get\_generators(CH_{k-1}, H_k)$ ;
17    end
18    else
19       $CH_{k-1} \leftarrow \{h \in H_{k-1} | h.closed = true\}$ ;
20       $Get\_generators(CH_{k-1})$ ;
21    end
22 end
23  $CH_k \leftarrow H_k$ ;
24  $Get\_generators(CH_k)$ ;
25  $CH \leftarrow \cup_k CH_k$  with generators;
26 Calculate utility unit array of each closed itemset;
27 End procedure
```

The pseudo-code of the HUCI-Miner algorithm, provided in Algorithm 1, is a level-wise procedure. It identifies all the high utility itemsets successively as the high utility closed itemset or generator in each set H_k , $1 \leq k \leq max$, contains the high utility itemset of length k . It derives the sets CH_k , $1 \leq k \leq max$, containing the closed itemsets and their supports, utility values and the corresponding generators. At first, it finds all the high utility itemsets using the FHM algorithm. After this exploration, the algorithm examines each high utility k -itemset, $k \geq 2$, which is a generator of a high utility k -itemset ($k \geq 2$) by considering the supports of all its subsets of length $k - 1$. The algorithm then verifies if it is a closed itemset by examining the supports of all its subsets of length $k - 1$. Two boolean variables *closed* and *key* are used in order to identify whether an itemset is a high utility closed itemset or a generator. If H_k is empty and H_{k-1} is nonempty, by consequence of Property 5 the elements of H_{k-1} are closed and it is performed in Steps 15-16. Conversely, if H_k is nonempty and H_{k-1} is empty, all itemsets in H_k are generators, and no extra step is needed as all itemsets are initially marked as generators.

An itemset c is identified as a generator during Steps 8-12 in the algorithm *HUCI-Miner*. If the support of c is same as one of its subsets having length $k - 1$ in H_{k-1} , then c is not a generator, and conversely, it is not closed. In Steps 15-18, all the closed itemsets of length $k - 1$ are added to the set CH_{k-1} . Step 21 discovers the closed itemset of the maximum length. In Steps 16, 19 and 22, the *Get_generators* procedure is called in order to update the global list of generators and assign the generators to the respective closed itemset. It takes the set CH_k as input. For each closed itemset

$ch \in CH_k$, its proper subsets in the global set of generators HG are then removed and then added to the generators of ch (Steps 1-5 in *Get_generators* procedure). This procedure updates the the global set of generators HG by the itemsets, which are not closed but they are generators before the starting of the next iteration. If the set of generators of a given closed itemset is empty, it indicates that the closed itemset is the generator of itself. For example, considering the database given in Table 1 with $min_util = 20\%$. The HUCIs are $G()$, $F()$, $E()$, $BF(B)$, $DE(D)$, $FE()$, $ACE(A)$, $DFE(DF)$, $BDFE(BE)$ and $ACFE(AFE)$, where $X(Y)$ means X is the closed itemset and Y is its generator.

Procedure: *Get_generators*(CH_{k-1}, H_k)

Input: CH_{k-1} : high utility closed itemset of length $(k - 1)$

Output: Assign the generators to each closed itemsets of CH_{k-1}

```

1 for each itemset  $ch \in CH_k$  do
2   for all subsets  $c' \in HG$  of  $ch$  do
3     add  $c'$  in  $ch.generator$ ;
4   end
5 end
6  $HG = HG \cup \{h \in H_k | h.key = true \wedge h.close = false\};$ 

```

Theorem 5. *For a given minimum utility threshold, the proposed HUCI-Miner algorithm generates all high utility closed itemsets with their generators correctly.*

Proof. The correctness of our proposed HUCI-Miner algorithm is based on Theorems 2 and 3. Theorem 2 determines a high utility itemset h_{k-1} , which is a high utility closed itemset, by comparing its support with the supports of the high utility k -itemset h_k containing the itemset h_{k-1} . Theorem 3 enables if a high utility k -itemset h_k is a generator by examining its support with the supports of the high utility $(k - 1)$ -itemsets, which are included in h_k . Since a high utility closed itemset can not be a generator for the large itemsets, they are not in the global list of generators. Again, the closure of a non-generator itemset c is the smallest superset of c in the set of the high utility closed itemsets. This proves that the identification of the generators of a closed itemset is correct. \square

5 Performance evaluation

In this Section we demonstrate the compactness of high utility itemsets by our proposed HUCI-Miner algorithm using both synthetic (T10I4D100K) and real datasets (Foodmart, Chess, Mushroom, Retail, Chain-store).

Table 5: Characteristics of datasets

Dataset	$ T $	$ I $	$AvgL$	$MaxL$	Type
Foodmart	4,141	1,559	4.4	14	Sparse
Chess	3,196	75	37	37	Dense
Mushroom	8,124	119	23	23	Dense
T10I4D100K	100,000	870	10.1	29	Sparse
Retail	88,162	16,470	10.3	76	Sparse
Chain-store	1,112,949	46,086	7.2	170	Sparse

The datasets T10I4D100K, Chess, Mushroom and Retail are obtained from frequent itemset mining dataset repository [12], Chain-store is obtained from NU-MineBench 2.0 [35], and Foodmart is from

Microsoft foodmart 2000 database. Table 5 shows the characteristics regarding these datasets in terms of the number of transactions ($|T|$), the number of distinct items ($|I|$), the average number of items in a transaction ($AvgL$), the maximum length of transaction $MaxL$, and its type: dense or sparse. Except Chain-store and Foodmart, the other four remaining considered datasets do not provide unit profits of each item (external utility) and item count for each transaction (internal utility). As in [2, 3, 28, 29, 45], we assign in each transaction of T10I4D100K, Mushroom, Chess and Retail, the internal utilities randomly between 1 and 10, and the external utilities of each item are randomly generated using a log-normal distribution between the range 0.01 and 10. The FHM implementation is downloaded from the SPMF framework [13]. We implement HUCI-Miner algorithm using the Java programming language and run with the Windows 7 operating system on a machine with CPU clock rate 3.0 GHz and intel core 2 quad processor with 3.18GB of main memory. To enhance the performance of all algorithms, the high utility itemsets are stored in the main memory. In the experiments we have used the minimum utility threshold min_util , as the percentage of total transaction utility values of the database.

Table 6: Number of high utility closed itemset (HUCI) of different datasets

Dataset	min_util in (%)	# of HUI	# of HUCI	# HG	# of HUCI+HG
Foodmart	0.07	637	605	22	627
	0.06	1483	770	301	1071
	0.05	6266	1076	1573	2649
	0.04	20766	1762	4686	6448
Chess	28	1982	1519	433	1952
	27	3874	2776	962	3738
	26	7187	4910	1937	6847
	25	13331	8700	3811	12511
Mushroom	8	28763	982	3253	4235
	7	51187	1470	5455	6925
	6	108797	2216	10161	12377
	5	222731	3308	17303	20611
T10I4D100K	0.009	93348	66572	9885	76457
	0.008	111172	75911	13050	88961
	0.007	141382	90187	18851	109038
	0.006	192673	111544	28837	140381
Retail	0.05	1876	1874	2	1876
	0.04	2810	2802	8	2810
	0.03	4848	4828	20	4848
	0.02	9341	9289	52	9341
Chain-store	0.08	118	118	—	118
	0.07	151	151	—	151
	0.06	193	193	—	193
	0.05	260	260	—	260

Table 6 reports the number of high utility itemsets (HUI), high utility closed itemsets (HUCI), and high utility generators (HG) extracted by HUCI-Miner in various datasets with different min_util values. The itemset, which is a generator of itself, is not counted in HG. The symbol ‘—’ represents that there is no generator except those, who are generators of themselves. From Table 6, we see that the total number of HUCI and HG is less than or equal to the total number of high utility itemsets. Since all HUIs can be generated from HUCIs with the help of utility unit array, HGs help in finding non-redundant rules

in support-confidence framework of high utility mining. Note that our proposed HUCI-Miner algorithm achieves a great reduction in compressing the number of high utility itemsets.

6 Conclusion

In this paper, we have first integrated the concept of minimal generator into high utility itemset mining. We have then shown that in order to mine minimal generators in high utility itemset mining, all high utility itemsets need to extract first and then to generate the minimal high utility generators. In order to achieve this, we have proposed a level-wise-search algorithm, HUCI-Miner, that identifies the high utility itemsets, high utility closed itemsets and associates the high utility generators to the corresponding closed itemsets. From the generated high utility closed itemsets with their generators, various condensed representations of association rules from the literature can be extracted from high utility itemsets.

References

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94*, pages 487–499, 1994.
- [2] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. Huc-prune: an efficient candidate pruning technique to mine high utility patterns. *Applied Intelligence*, 34(2):181–198, 2011.
- [3] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. Efficient tree structures for high utility pattern mining in incremental databases. *Knowledge and Data Engineering, IEEE Transactions on*, 21(12):1708–1721, 2009.
- [4] J. L. Balcázar. Formal and computational properties of the confidence boost of association rules. *ACM Trans. Knowl. Discov. Data*, 7(4):19:1–19:41, 2013.
- [5] B. Barber and H. J. Hamilton. Parametric algorithms for mining share frequent itemsets. *Journal of Intelligent Information Systems*, 16(3):277–293, 2001.
- [6] B. Barber and H. J. Hamilton. Extracting share frequent itemsets with infrequent subsets. *Data Mining and Knowledge Discovery*, 7(2):153–185, 2003.
- [7] C. H. Cai, A.W.-C. Fu, C.H. Cheng, and W. W. Kwong. Mining association rules with weighted items. In *Database Engineering and Applications Symposium, 1998. Proceedings. IDEAS'98. International*, pages 68–77, 1998.
- [8] C. Carter, H. J. Hamilton, and N. Cercone. Share based measures for itemsets. In *Principles of Data Mining and Knowledge Discovery*, volume 1263 of *LNCS*, pages 14–24. Springer Berlin Heidelberg, 1997.
- [9] R. Chan, Q. Yang, and Y.-D. Shen. Mining high utility itemsets. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 19–26, 2003.
- [10] J. Cheng, Y. Ke, and W. Ng. Effective elimination of redundant association rules. *Data Min. Knowl. Discov.*, 16(2):221–249, 2008.

- [11] A. Erwin, R P. Gopalan, and N. R. Achuthan. Efficient mining of high utility itemsets from large datasets. In *Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'08, pages 554–561, 2008.
- [12] FIMI. Fimi: The frequent itemset mining dataset repository, 2003. URL <http://fimi.cs.helsinki.fi/data/>. Accessed on February 2012.
- [13] P. Fournier-Viger, A. Gomariz, A. Soltani, and T. Gueniche. SPMF: Open-Source Data Mining Library, 2014. URL <http://www.philippe-fournier-viger.com/spmf/>. Accessed on August 2014.
- [14] P. Fournier-Viger, C.W. Wu, S. Zida, and V. S. Tseng. Fhm: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In *Foundations of Intelligent Systems*, volume 8502 of *LNCS*, pages 83–92. Springer International Publishing, 2014.
- [15] L Geng and H J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), 2006.
- [16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.
- [17] R. J. Hilderman. Predicting itemset sales profiles with share measures and repeat-buying theory. In *Intelligent Data Engineering and Automated Learning*, volume 2690 of *LNCS*, pages 789–795. Springer Berlin Heidelberg, 2003.
- [18] R. J. Hilderman, C. L. Carter, H. J. Hamilton, and N. Cercone. Mining market basket data using share measures and characterized itemsets. In *Research and Development in Knowledge Discovery and Data Mining*, volume 1394 of *LNCS*, pages 159–173. Springer Berlin Heidelberg, 1998.
- [19] R. J. Hilderman, H. J. Hamilton, C. L. Carter, and N. Cercone. Mining association rules from market basket data using share measures and characterized itemsets. *International Journal on Artificial Intelligence Tools*, 07(02):189–220, 1998.
- [20] Y. Koh, R. Pears, and W. Yeap. Valency based weighted association rule mining. In *Advances in Knowledge Discovery and Data Mining*, volume 6118 of *LNCS*, pages 274–285. Springer Berlin Heidelberg, 2010.
- [21] M. K. Kryszkiewicz. Representative association rules. In *PAKDD*, volume 1394 of *LNCS*, pages 198–209. Springer, 1998.
- [22] G.-C. Lan, T.-P. Hong, and V. S. Tseng. An efficient projection-based indexing approach for mining high utility itemsets. *Knowledge and Information Systems*, pages 1–23, 2013. doi: 10.1007/s10115-012-0492-y.
- [23] H.-F. Li, H.-Y. Huang, Y.-C. Chen, Y.-J. Liu, and S.-Y. Lee. Fast and memory efficient mining of high utility itemsets in data streams. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 881–886, 2008.
- [24] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. A fast algorithm for mining share-frequent itemsets. In *Web Technologies Research and Development - APWeb 2005*, volume 3399 of *LNCS*, pages 417–428. Springer Berlin Heidelberg, 2005.

- [25] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Direct candidates generation: A novel algorithm for discovering complete share-frequent itemsets. In *Fuzzy Systems and Knowledge Discovery*, volume 3614 of *LNCS*, pages 551–560. Springer Berlin Heidelberg, 2005.
- [26] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. *Data & Knowledge Engineering*, 64(1):198 – 217, 2008.
- [27] C-W. Lin, T-P. Hong, G-C. Lan, J-W. Wong, and W-Y. Lin. Incrementally mining high utility patterns based on pre-large concept. *Applied Intelligence*, pages 1–15, 2013. doi: 10.1007/s10489-013-0467-z.
- [28] M. Liu and J. Qu. Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 55–64, 2012.
- [29] Y. Liu, W.-K. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In *Proceedings of the 1st International Workshop on Utility-based Data Mining, UBDM '05*, pages 90–99, 2005.
- [30] J.S. Park, M.-S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD Rec.*, 24(2):175–186, 1995.
- [31] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Inf. Syst.*, 24(1):25–46, 1999.
- [32] N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal. Generating a condensed representation for association rules. *J. Intell. Inf. Syst.*, 24(1):29–60, 2005.
- [33] R. Pears, Y. S. Koh, G. Dobbie, and W. Yeap. Weighted association rule mining via a graph based connectivity model. *Information Sciences*, 218(0):61 – 84, 2013.
- [34] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. In *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM '01*, pages 441–448, 2001.
- [35] J. Pisharath, Y. Liu, W. K. Liao, A. Choudhary, G. Memik, and J. Parhi. Nu-minebench version 2.0 dataset and technical report, 2005. URL <http://cucis.ece.northwestern.edu/projects/DMS/MineBench.html>. Accessed on June 2013.
- [36] G. D. Ramkumar, S. Ramkumar, and T. Shalom. Weighted association rules: Model and algorithm. In *Proc. Fourth ACM Intl Conf. Knowledge Discovery and Data Mining*, 1998.
- [37] J. Sahoo, A. K. Das, and A. Goswami. An effective association rule mining scheme using a new generic basis. *Knowledge and Information Systems*, pages 1–30, 2014. doi: 10.1007/s10115-014-0732-4.
- [38] B.-E. Shie, V. S. Tseng, and P. S. Yu. Online mining of temporal maximal utility itemsets from data streams. In *Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10*, pages 1622–1626, 2010.
- [39] W. Song, Y. Liu, and J. Li. Mining high utility itemsets by dynamically pruning the tree structure. *Applied Intelligence*, pages 1–15, 2013. doi: 10.1007/s10489-013-0443-7.

- [40] K. Sun and F. Bai. Mining weighted association rules without preassigned weights. *Knowledge and Data Engineering, IEEE Transactions on*, 20:489–495, 2008.
- [41] L. Szathmary, A. Napoli, and S.O. Kuznetsov. Zart: A multifunctional itemset mining algorithm. In *In: Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA 2007)*, pages 26–37, 2007.
- [42] L. Szathmary, P. Valtchev, A. Napoli, R. Godin, et al. An efficient hybrid algorithm for mining frequent closures and generators. In *6th International Conference on Concept Lattices and Their Applications (CLA'08)*, pages 47–58, 2008.
- [43] L. Szathmary, P. Valtchev, A. Napoli, and R. Godin. Efficient vertical mining of frequent closures and generators. In *Advances in Intelligent Data Analysis VIII*, pages 393–404. Springer, 2009.
- [44] F. Tao, F. Murtagh, and M. Farid. Weighted association rule mining using weighted support and significance framework. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 661–666, 2003.
- [45] V. S. Tseng, C.-W. Wu, B.-E. Shie, and P. S. Yu. Up-growth: An efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 253–262, 2010.
- [46] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1772–1786, 2013.
- [47] W. Wang, J. Yang, and P. S. Yu. Efficient mining of weighted association rules (war). In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 270–274, 2000.
- [48] G.I. Webb. Discovering significant rules. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2006)*, pages 434 – 443. ACM, 2006.
- [49] C. W. Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng. Efficient mining of a concise and loss-less representation of high utility itemsets. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 824–833, 2011.
- [50] C. W. Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng. Efficient algorithms for mining the concise and lossless representation of closed+ high utility itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 99(Preliminary):1, 2014. doi: <http://doi.ieeecomputersociety.org/10.1109/TKDE.2014.2345377>.
- [51] Y. Xu, Y. Li, and G. Shaw. Reliable representations for association rules. *Data & Knowledge Engineering*, 70(6):555–575, 2011.
- [52] S.B. Yahia, G. Gasmi, and E.M. Nguifo. A new generic basis of “factual” and “implicative” association rules. *Intell. Data Anal.*, 13(4):633–656, 2009.
- [53] H. Yao, H. J. Hamilton, and C. J. Butz. A foundational approach to mining itemset utilities from databases. In *Proceedings of the Third SIAM International Conference on Data Mining*, pages 482–486, 2004.

- [54] H. Yao, H. J. Hamilton, and L. Geng. A unified framework for utility-based measures for mining itemsets. In *Proceedings of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining*, pages 28–37, 2006.
- [55] U. Yun. Efficient mining of weighted interesting patterns with a strong weight and/or support affinity. *Information Sciences*, 177(17):3477 – 3499, 2007.
- [56] M.J. Zaki. Mining non-redundant association rules. *Data Min. Knowl. Discov.*, 9(3):223–248, 2004.